

### REMARKS

Claims 1, 3 to 14, and 16 to 28 are pending in this application. Of these, claims 1 and 14 are independent.<sup>1</sup> Favorable reconsideration and further examination are respectfully requested.

Claim 14 has been amended to change “information carrier” to “tangible computer-readable medium”. Accordingly, withdrawal of the §101 rejection is respectfully requested.

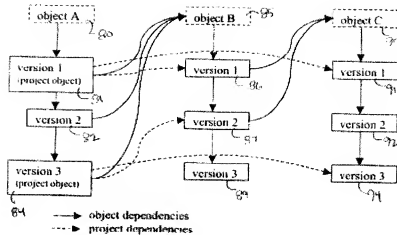
All of the claims were rejected over U.S. Patent Publication No. 2002/0178181 (Subramanyan) in view of U.S. Patent Publication No. 2004/0002049 (Beavers). This rejection is respectfully traversed for at least the following reasons.

Independent claim 1 recites that the version dependency data identifies versions of other learning objects upon which the project object depends, and that the other learning objects, including the first and second objects, do not store version dependency data. The other learning objects store dependency data that identifies an object dependency but that does not identify a version dependency, and the other learning objects rely on the version dependency data in the project object for identification of version dependency. Claim 1 thus recites different terms, namely “dependency data” and “version dependency data”, the combination of which are not found in the cited references.

An embodiment of claim 1 is shown in Fig. 8 of the application, which is reproduced below.

---

<sup>1</sup> The Examiner is urged to independently confirm this recitation of the pending claims.



As explained in our previous responses, a project object 81 stores version dependency data, which is identified by the dashed lines. The version dependency data identifies the versions of other objects, here object 85 and object 90, upon which the project object 81 depends. In this example, object 85 exists in three versions: 86, 87 and 89; and object 90 exists in three versions: 91, 92 94. The version dependency data identifies the version of the object upon which the project object depends. A first object, e.g., object 85, contains dependency data, which is identified by solid lines. The dependency data identifies a second object, e.g., object 90, upon which the first object 85 depends, but does not identify the version upon which object 90 depends. The version dependency data is stored in the project object. The other learning objects therefore rely on the project object for identification of version dependency.

It was said in the Office Action that:

With respect to claim 1,

**Subramanyan discloses a method, performed by one or more processing devices, for use in an electronic learning system that stores information as learning objects, the method comprising: designating a target learning object as a project object**

**and storing version dependency data in the project object, the version dependency data identifying versions of other learning objects upon which the project object depends, the other learning objects including at least a version of a first object upon which the project object directly depends, and a version of a second object upon which the project object indirectly depends the project object being an object that is separate from the first object and second object; wherein the other learning objects including the first and second objects do not store version dependency data and wherein the other learning objects store dependency data that identifies an object dependency but that does not identify a version dependency, the other learning objects relying on version dependency data in the project object for identification of version dependency (paragraphs 9,12, 16, 19 – 21, 31 and 33, Subramanyan).**

**Subramanyan does not explicitly disclose the object and version dependency as claimed.**

2

We disagree with these statements in the Office Action. More specifically, Subramanyan says very little about its version control tool, and certainly nothing that would imply that Subramanyan's versions/version control tool have/has the structure ascribed to it by the Office Action. In this regard, Subramanyan's version control tool is only mentioned in the following paragraphs of Subramanyan, with specific discussion highlighted in bold and underlined:

[0016] It is also an object of the present invention to provide a method and system of the above character including developing a storyboard server which has a

---

<sup>2</sup> Office Action, pages 3 and 4

**built-in version control that enables team members and users to view changes in their work and revert back to older versions.**

[0031] The storyboard server comprises a number of enabling software tools and databases, including a storage database and file system, a templating tool, an authoring tool, a page level storyboarding tool, a messaging tool, **a version control tool**, and a synchronous communication tool. The storage database and file system allows users to upload, download, copy, etc., multiple media files to and from the storyboard server. The files may be saved in the database, organized in the file system, and accessed by users when necessary.

[0035] Each of these tools comprising the storyboard server, the templating, authoring, and page level storyboarding tools, work in conjunction with other storyboard tools to facilitate and integrate the storyboarding process, including a messaging tool, **a version control tool**, and a synchronous communication tool. The messaging tool enables users to communicate with each other. For example, if a template is created by an instructional designer and ready for use or completion by a subject matter expert, the instructional designer need only create the template in the storyboard and indicate that it is ready. The storyboard messaging tool will then automatically send a message, such as by e-mail, to the subject matter expert indicating that the template is ready for his or her attention. The messaging tool may similarly send automatic e-mail or other messages to users at various points in the storyboarding process for any number of reasons.

[0036] **The version control tool enables users to access, edit, copy, etc., previous versions of work contained in the storyboard server, including templates, empty learning objects and learning paths, frames, pages, and edited content or subject matter, as well as any textual information.**

[0040] Each storyboard of this embodiment of the present invention comprises audio and visual segments or elements for one screen. There are means within the storyboard for expressing narrative text for the audio segment or elements and text for display of the visual segment or elements. The storyboard may also include multiple audio narratives or multiple visual elements on one screen. The storyboard additionally comprises means for specifying the sequences in time for the rendering or utilization of audio and visual elements. These timing means allow the audio and visual segments to be played out synchronously. **The storyboard also comprises messaging and tracking or version control.**

[0048] Users, including content developers, may post questions 22 regarding, for example, learning objects or paths or frames, which other users will receive. Collaboration continues between and among the users and developers to provide

and receive feedback on the project and any questions asked 28. The parties requested for assistance will then provide feedback to the content developers 29. The subject matter experts may have 23 and provide 25 feedback on the content as well. Project managers and other users can mark the subject matter experts' feedback as a defect, enhancement, or change, or other customized designation. This designation process may assist the project manager in costing changes that result from feedback. As templates, learning objects, and frames are modified, a history of these items is stored in the storyboard server. Users can revert back to previous versions. When the users and developers sign off on the content, the content is ready to go 24.

[0056] The PM takes all the screens within the learning objects and assigns them to the designers. Working remotely, the designers log into the Storyboard Server, get detailed information on the screens they are to produce, and upload the finished product. The SME also logs in and reviews the screens as they are being finished. The screens look great, but there are a few things the SME wants changed. She adds comments for the screens and the designers make the changes. At one point, the SME wants to go back to an earlier version of a screen where the new changes didn't quite work right. The Storyboard Server's version control makes this possible.

Except for the claims, the entire description of Subramanyan's version control tool is set forth above in bold and underlining. As you can see, Subramanyan's version control involves storing a history of on-screen items, and reverting to that history, if necessary. There is no description whatsoever in Subramanyan of storing version dependency data or dependency data in an object, much less a project object, as claimed. There is also no description in Subramanyan of using version dependency data or dependency data to identify object dependencies, as claimed. For example, there is no description in Subramanyan of using data in a project object for identification of version dependency. In fact, Subramanyan's system is so entirely different from what is claimed that, respectfully, we are at a loss to understand how the allegations regarding Subramanyan on pages 3 and 4 of the Office Action can be made.

The Office Action further states:

Subramanyan does not explicitly disclose the object and version dependency as claimed.

Beavers, however teaches the object and the version dependency as claimed in paragraphs 149 and 174.

3

We disagree with this characterization of Beavers. The cited paragraphs of Beavers are reproduced below, with the specific discussion highlighted in bold and underlined:

[0149] The next option category in the options list 800 is the "Properties" category 806. Upon selection of the properties option category 806, a dialog box (not shown) appears over a portion of the slideshow window. This dialog box is used in the typical manner by the presenter to specify certain network parameters associated with the broadcast of the presentation. Specifically, a standard data entry field is included for the presenter to enter the desired packet delay, which could be set between 0 and 30 milliseconds in tested versions of the learning system (although 20 milliseconds was used as a default setting). In addition, a forward error correction (FEC) data entry field is included in the dialog box. **In tested versions of the learning system, the forward error correction took the form of a simple process of broadcasting multiple versions of the presentation data in case a packet of any one of the repeated broadcast is lost or corrupted during transit over the network. The aforementioned data entry field is used to specify how many times a packet of the presentation data is to be sent. In tested versions of the system this could be varied between 1 and 5 times, with 2 being the default.**

[0174] The last menu item is the help item 534. Selecting the help menu item 534 causes a drop-down list (not shown) to appear in the slideshow window. **In tested versions of the present system, this list included only one listing--namely an "About" link.** Selecting the about link causes an information box to appear in the slideshow window. This information box is a standard "about" box and includes information such as the title of the program and **its version number**, copyright data, the licensee's name, and perhaps links to other informational windows. While the tested embodiments were limited to just the about link, this need not be the case. Links to various help directories, as well as links to "on-line" support sites, could be included.

---

<sup>3</sup> Office Action, page 4

Again, respectfully, we do not at all see a correlation between the cited paragraphs of Beavers and what is claimed. Regarding the dependency data and the version dependency data, claim 1 recites the following:

the version dependency data identifying versions of other learning objects upon which the project object depends

the other learning objects store dependency data that identifies an object dependency but that does not identify a version dependency, the other learning objects relying on version dependency data in the project object for identification of version dependency.<sup>4</sup>

Where are these features found in the cited paragraphs of Beavers, or elsewhere in Beavers for that matter? As we understand it, Beavers includes different system versions, included tested versions. Beavers also describes “broadcasting multiple versions of the presentation data in case a packet of any one of the repeated broadcast is lost or corrupted during transit over the network”. We do not understand how this information could correlate to the dependency data and the version dependency data that is claimed. For example, the version dependency data, as claimed, identifies versions of other learning objects upon which the project object depends. None of the Beavers data is indicated to identify object dependencies. The dependency data, as claimed, identifies an object dependency but that does not identify a version dependency. There is no indication that any version data in Beavers (to the extent that there is any) identifies an object dependency but not a version dependency.

---

<sup>4</sup> Claim 1

Given what we believe to be the glaring deficiencies of both Subramanyan and Beavers, even if the two references were combined, the resulting hypothetical combination would fail to render claim 1 obvious. Accordingly, claim 1 is believed to be patentable.

Amended independent claim 14 is a computer program product claim that roughly corresponds to independent claim 1, and is also believed to be patentable.

Dependent claims are also believed to define patentable features of the invention. Each dependent claim partakes of the novelty of its corresponding independent claim and, as such, each has not been discussed specifically herein.

It is believed that all of the pending claims have been addressed. However, the absence of a reply to a specific rejection, issue or comment does not signify agreement with or concession of that rejection, issue or comment. In addition, because the arguments made above may not be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper, and the amendment of any claim does not necessarily signify concession of unpatentability of the claim prior to its amendment.

In view of the foregoing amendments and remarks, we respectfully submit that the application is in condition for allowance, and such action is respectfully requested at the Examiner's earliest convenience.

The undersigned attorney can be reached at the address shown below. All telephone calls should be directed to the undersigned at 617-521-7896.



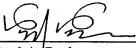
Applicants : Wolfgang Theilmann, et al.  
Serial No. : 10/809,873  
Filed : March 25, 2004  
Page : 19 of 19

Attorney's Docket No.: 13909-161001  
Client Ref.: 2004P00116US

Please apply any fees or credits due in this case to Deposit Account 06-1050 referencing  
Attorney Docket No. 13909-161001.

Respectfully submitted,

Date: August 12, 2008

  
\_\_\_\_\_  
Paul A. Pysher  
Reg. No. 40,780

Fish & Richardson P.C.  
225 Franklin Street  
Boston, MA 02110-2804  
Telephone: (617) 542-5070  
Facsimile: (617) 542-8906

22000331.doc